

On the Empirical Complexity of Text Classification Problems

Omid Madani
AI Center, SRI International
333 Ravenswood Ave
Menlo Park, CA 94025

Hema Raghavan Rosie Jones
Yahoo! Inc.
4 Cambridge Center, 11th floor.
Cambridge, MA 02142

SRI Artificial Intelligence Center

Technical Report 567
September 23rd, 2009

Abstract

In order to train a classifier that generalizes well, different learning problems, in particular high-dimensional ones such as text classification, can require widely different amounts of training, as measured in terms of the number of training instances required to reach adequate accuracy or the number of features effectively utilized in the classifier. We define several measures of learning difficulty and explore their utility in approximately capturing the inherent complexity of text classification problems. These measures can be efficiently computed for real-world problems for which linear classifiers are effective. We observe an intimate relationship (a high positive correlation) between feature complexity and instance complexity when using the measures.

Such measures of difficulty are useful for comparing learning problems and corpora, and gaining insight into the variable success of methods such as active learning. In particular, we quantify the difficulty of 358 text classification problems and 9 corpora using the proposed measures, including those popularly used for benchmarking the performance of text classification algorithms, such as the Reuters and 20 Newsgroups corpora. We demonstrate the spectrum of problems that exist in text classification in addition to quantifying results that have only been qualitatively discussed in the text classification literature. We observe that many problems in the commonly used data sets are of low to medium complexity, that is, only roughly tens of well-selected features are required to gain most of the maximum attained performance on such concepts, when using linear classifiers. We find that learning for such types of problems especially stands to benefit from incorporating feature feedback (prior knowledge on features) into active learning techniques.

1 Introduction

During our work in text classification we have observed that some tasks are considerably more complex than others. For a relatively simple task, a classifier that checks for only a few features may effectively classify all documents. An example of a simple task is that of classifying *wheat* documents in the Reuters corpus [2]. Boolean expressions that can classify a *wheat* document in the Reuters corpus are shown in Table 1. In fact for this task, just checking for the occurrence of the term *wheat* in the document gives a high level of accuracy. Other tasks, for example classifying documents by genre, are not as simple and the resulting expressions for classification involve a large number of features.

| | | |
|-----------------------------------|---|-------|
| <i>wheat & farm</i> | → | WHEAT |
| <i>wheat & commodity</i> | → | WHEAT |
| <i>bushels & export</i> | → | WHEAT |
| <i>wheat & agriculture</i> | → | WHEAT |
| <i>wheat & tonnes</i> | → | WHEAT |
| <i>wheat & farm&¬soft</i> | → | WHEAT |

Table 1: Induced rule set using the CONSTRUE system for categorizing *wheat* documents in the Reuters data set. The induced rules result in 99% accuracy.

One view of concept complexity or difficulty can be associated with the *value* of the maximum achievable accuracy; that is, a concept that cannot be learned to a desired degree of accuracy may be considered to be a difficult one. Studying difficulty from that perspective is important in itself, but is not the goal of this work. In this work, we restrict ourselves to concepts that we know are ultimately adequately learnable by the chosen algorithm (in our case, robust algorithms for learning linear classifiers) and ask how quickly they can be learned. In this regard, our work can be viewed as an empirical analogue of much work in theoretical machine learning in which the *sample complexity* of learning of a concept class under consideration is studied [6, 26]. In such work, the assumption is often that the class of concepts in question contains a target concept (or concepts) that performs well, and one major question is on the number of instances needed, asymptotically, to learn (a sufficient approximation of) the target concept. Here, we explore definitions and *empirically computed* approximations for instance as well as feature complexities. For many text classification tasks, it has been found that robust linear classifiers such as support vector machines (SVMs) work quite well [25, 17, 59, 62, 33], given that it is typically possible to utilize a fairly large feature space. We use this finding and focus on linear classifier learning methods in exploring problem difficulty. The proposed measures approximate the empirical difficulty of concepts in commonly used data sets, illustrating the spectrum of problems that exist in text classification. We find that concepts that can be learned using fewer examples can be described by a few well chosen features, and vice versa.

Given a (linear classifier) learning algorithm, a set of features and a concept, there is some maximum achievable measure of categorization accuracy ($\leq 100\%$) that the learner can achieve in the limit. For example, even if the data is not exactly linearly separable, a linear SVM may be able to achieve some fairly reasonable and acceptable accuracy (often in the order of 90% for many text categorization problems) with adequate training data. Given such a set of concepts that are “almost linearly separable”, that is “learnable” by a robust method such as a linear SVM, we ask “how many features” are sufficient to obtain nearly maximum achievable accuracy. Our (theoretically ideal) *feature complexity* scores are based on the size of the minimum set of features sufficient for achieving a desired accuracy level. In this case, all the training instances are available (except for test instances). Likewise, our *instance complexity* scores are based on the minimum size set of instances needed for achieving a desired accuracy level, where all the features of the problem are available to the learner. The idealized measures involve subset selection, and their exact computation is practically very expensive or simply intractable. Instead, we aim to only approximately capture the inherent sample-size and feature-size complexities of problems, and we provide evidence for their usefulness.

Our instance complexity measures are designed to capture the minimum number of training examples needed to attain nearly maximum achievable accuracy by the learner, when all the features are available. Thus, the training examples need not be selected at random; they can be intelligently picked. We utilize active learning for instance selection. A problem for which training on a few well-picked instances is sufficient to arrive at the maximum achievable accuracy is a low instance complexity problem. Analogous to instance complexity, we define feature complexity with the goal of (approximately) capturing the minimum number of intelligently picked features needed to achieve nearly maximum possible accuracy, when ample training instances are available. If a concept can be described by a weighted combination of a few well-selected features, it is considered to be of low feature complexity. Very importantly, we take logarithms of the feature set size and instance set size in measuring complexity, as we explain, to emphasize relative differences and to lower dependencies on the particular choices of instance and feature selection methods.

We note that our measures can be interpreted as *upper estimates* of complexity, since it is impractical to test all instance and feature selection algorithms. No feature selection or active learning algorithm is universally the best. Due to the limits of the specific algorithms that we use, it is likely that for many of the problems that we test, one can obtain lower feature or instance complexity scores by using other algorithms, in particular for reaching a specific accuracy target (e.g., 90% F1 score). However, upper estimates can still be useful, especially when one uses readily available and commonly used efficient methods, such as perceptron learning. For example, one can still make sound observations or statements such as, “many experimental learning problems are relatively easy”, in that it is observed that many such problems require relatively few well-selected features (eg tens of features) or instances (and one can make explicit which methods were used to obtain the scores, as we do). In our experiments, we compare complexity scores obtained using a few different learning algorithms and instance-selection methods, but our experiments are not exhaustive.

In obtaining the proposed complexity scores of specific learning problems, we have made certain approximations and, furthermore, the measures are relative (e.g., to the methods and features used). This is not unlike the setting of the asymptotic analysis of algorithms, wherein one uses the big-O notation and additive and multiplicative constants are usually ignored, and an assumption is made of the type of the problem instances encountered, such as the worst-case assumption. While such measures are not perfect and their use results in some loss of precision, they provide an abstraction that has found ample utility in comparing problems and algorithms, and in designing new algorithms. We provide some evidence for the utility of the measures we propose and explore, that we summarize below.

We study the relationship between the measures in text classification, a domain of high dimensionality with many relevant and irrelevant features. We find that instance complexity and feature complexity are highly positively correlated and yield similar rankings of problems and corpora (Section 5.1). This observation is consistent with the intuition that problems requiring large numbers of instances tend to require large numbers of features, and vice versa. If a problem requires a large number of instances for robust linear learning algorithms, it is likely the case that appropriate weights for a relatively large number of features need to be estimated.¹ Conversely, if for good accuracy, computing appropriate weights for a sizable number of features is required, then the number of learning instances required, to identify such features and compute and validate a good weighting, would be relatively high. The high positive correlation also provides evidence that the proposed measures indeed (approximately) capture the inherent feature and instance complexity of a problem.

We benchmark 9 corpora and 358 text classification problems for their difficulty (Table 3), and analyze and discuss the reasons for the differences in complexity in several cases. Such knowledge can aid researchers and practitioners in assessing their own learning problems or in the selection of commonly used test data sets, and in anticipating learning performance. We find that many categorization problems are in the low to mid range complexity, that is, the number of intelligently picked features sufficient to obtain most of the accuracy, in terms of precision and recall, is in the tens. Such findings can help us develop a perspective on the difficulty of text classification problems that tend to be encountered in practice.

We began this work by asking why prior knowledge on features was more useful in accelerating (plain active) learning for certain problems than others [15, 45, 58, 52, 44], and we considered whether there would be aspects of a problem that could explain or correlate well with the improvement from using feature knowledge. In Section 6 we describe how we use these measures to obtain insights into the kinds of text classification problems for which feature knowledge/feedback (in addition to document feedback) is especially useful. In particular, we find that problems with low to medium feature complexity stand to benefit most from feature feedback, and it is encouraging to

¹Other factors affect the complexity, in particular the proposed complexity scores, as well. For instance, the total number of features available also affects the convergence rates or sample complexity of various linear classifier learning algorithms. We demonstrate only relatively high positive correlations.

see that many experimental problems fall in this range (Table 3 and Figure 8).

In summary, in this paper we

- define a set of efficiently computable empirical measures of complexity of linear classification problems, and discuss the challenges in defining such empirical measures
- use these measures to understand the relationship between feature and instance complexity in linear text classification problems
- obtain insights on the range and the spread in difficulty of various problems in the available data sets, according to the proposed measures
- help identify problems for which interactive learning and use of feature prior knowledge are especially useful, and identify problems for which demonstrating an improvement in active learning performance is especially valuable

This paper is organized as follows. We describe the data sets in Section 2. We describe the complexity measures in Section 3 and the methods that we use to instantiate the measures in Section 4. Section 5 presents a variety of experiments on evaluating the measures as well as assessing the difficulty of problems and corpora utilizing those measures. Section 6 presents experiments relating problem complexity to success of active training with feature feedback. Section 7 presents related work and includes a discussion of other candidate complexity measures and complexity factors. Section 8 concludes.

2 Data

As mentioned in the previous section we specifically focus on text classification problems. We consider 9 corpora and 358 binary classification problems as shown in Table 2. Most corpora have topic-based category labels, except for three: (1) the Topic Detection and Tracking corpus that contains classes based on events, (2) the British National Corpus (BNC) where the classes are based on genre, and (3) the documents in the Enron corpus, which are emails categorized into folders by the recipient of the email. In computing complexity for the Reuters-RCV1 corpus we only used the 23149 training documents for efficiency reasons [31].

For all data sets we used unigram features. For some of them we further added n-grams of features if these n-grams improved performance. All words in the text were stemmed, and stop-words were removed using the rainbow toolkit [39]. Since we are interested only in measuring the difficulty of “learnable concepts”, we considered only those problems for which there was ample training data to achieve an acceptable level of performance (of above 75% Maximum F1) using a linear SVM (see Section 4.4 on measuring accuracy). The last column in Table 2 lists the average maximum $F1$ obtained using a linear classifier and bag-of-words features trained on a random sample comprising 90% of the data and tested on the remaining.

| Corpus | Domain | M | N | # topics | MaxF1 |
|--|-------------------------|-------|--------|----------|-------|
| Reuters-21578 [30] | News-wire | 9410 | 33378 | 10 | 0.874 |
| RCV1 [31] | News-wire | 23149 | 47236 | 87 | 0.759 |
| Topic Detection and Tracking (TDT) [1] | News-wire and broadcast | 67111 | 85436 | 10 | 0.918 |
| British National Corpus [8] | News, journals, etc. | 2642 | 233288 | 15 | 0.774 |
| Enron [4] | E-mail folders | 1971 | 711815 | 8 | 0.887 |
| 20 Newsgroups [29] | Newsgroup postings | 19976 | 137728 | 20 | 0.851 |
| Industry Sector [38] | Corporate web pages | 9565 | 69297 | 104 | 0.909 |
| TechTC-100 [14] | ODP hierarchy | 149 | 18073 | 100 | 0.972 |
| WebKB [12] | University websites | 2101 | 28682 | 4 | 0.918 |

Table 2: For all corpora, except TechTC-100, each learning problem (topic or category learning) is a single one-versus-rest binary classification problem. The TechTC-100 dataset consists of 100 binary classification problems with about 149 documents in each (about 50-50 split), and an average of 18073 features in each. M is the total number of instances and N is the total number of features in the corpus.

Some data sets have multiple versions and we have selected a subset of the classes or instances for some of the data sets. It is important that our results be understood within this relative context. For instance, researchers have used several versions of Reuters-21578, and we are using the version confined to the top 10 most frequent categories. See [16] for a detailed description of the versions.² We also comment that our complexity measures are most appropriate for those classes with relatively large numbers of positive examples (at least in the order of tens), so that test set accuracy can be measured effectively, and that we may have some reason to expect that the accuracy performance has somewhat stabilized.

3 Measures of complexity

We now describe six measures of complexity – three measuring instance complexity and three measuring feature complexity. Given a "learnable concept" (or an "almost linearly separable con-

²With labeled documents for other roughly 80 categories, the number of instances grows to almost 13000 instead of our roughly 10000.

cept”) with M labeled examples from which to estimate complexity, each represented as an N dimensional vector, our complexity measures quantify the difficulty of learning by measuring how many of the M instances and N features are really required to learn a good classifier.³

Our instantiation of these instance complexity measures attempts to capture *roughly* how many of the best (most informative) instances for a given problem are needed in order to achieve performance close to that of a linear classifier that has access to all the features and ample training examples. In computing instance complexity we use active learning (selective sampling) methods. Because such methods do not necessarily find the minimal set of required instances, we obtain an empirical overestimate on the ideal instance complexity. The tightness of the estimate is dependent on the active learning method used. The measure is an overestimate with respect to achieving the desired *fixed* accuracy. The number of required instances can increase if the desired accuracy is raised. Similarly, our feature complexity measures quantify roughly how many of the most informative features are needed to achieve close to the best accuracy reachable by the chosen classifier. Our feature complexity measures are also overestimates on the true feature complexity, where the tightness of the estimate is dependent on the feature selection method used.

In measuring performance and complexity, we use a logarithmic scale (e.g., log of the number of instances). The log scale captures the desired property that the difference between a pair of problems that respectively require 50 and 100 instances for highest accuracy is more intense than between two problems requiring 1000 and 1050 instances.⁴ We would like to say, for example, that two problems that have about 9 and 10 instance complexities (n_i) are somewhat similar in difficulty, while a problem with instance complexity of 6 is significantly easier. We discuss the approximate and relative nature of the proposed measures throughout the paper (and see Section 7).

3.1 Instance Complexity Measures

Given a classification algorithm and a binary classification problem, there is some maximum achievable performance often under 100% in practice (Table 2). As a measure of accuracy we use the (max) F1 score, which is the harmonic mean of precision and recall [56, 53] (and see Section 4.4). We denote by $F1(p, q)$ the maximum accuracy (in particular MaxF1 score on test data, see Section 4.4) achieved by the learner when it has access to the best set of p instances, from the training set, and the best set of q features (*i.e.*, the maximum is taken over all possible sets of certain size p and q), $p \leq M$ and $q \leq N$. We assume that the near best performance is achieved when the learner has access to all the available instances and features, $F1(M, N)$, and not for example a subset of the features. This assumption is a mild one in our setting: as will be seen, we will use only the approximate maximum accuracy that is empirically achieved using the learner, and

³Of course, the complexities could be measured for achieving relatively low accuracies (e.g., 50%) as well. The basic task is computing the resource requirements (instances or features) needed to achieve some performance level of interest.

⁴Akin to the Richter scale: an earthquake of magnitude 6 is significantly more intense than one of magnitude 5.

we need only an approximate number of instances to achieve this accuracy. Furthermore, for text classification problems, and in general linear classification problems, when one uses robust learners such as SVMs together with appropriate regularization, and where there is sufficient training data, the poor effects of many correlated, redundant, or noisy features are somewhat ameliorated⁵ [35, 36, 25, 18, 47]. We further discuss this aspect in related work (Section 7), in the context of feature selection.

In measuring the rate of learning we want to measure the minimum number of training examples (\hat{i}) needed to achieve the best performance for a given classifier. The brute-force way to find this minimum for a data set with M examples would involve training the classifier for every possible subset of training examples, that is, 2^M times. This is prohibitive for most M . Instead we use active learning to give us an ordering on the instances and compute an overestimate on \hat{i} , using this ordering to pick a subset, as we explain next.

Our active learning regime begins with 2 randomly selected instances, one in the positive and one in the negative class (a positive and a negative instance). The active learner learns a classifier based on this information and then intelligently chooses the next instance from a *pool* of unlabeled examples for the expert to label. The classifier is retrained and the process continues. We measure the performance, $F1(2^t, N)$, of the classifier, estimated on the held-out set, after every 2^t iterations of active learning with t varying as $1, 2, \dots, \log_2 M$, where M is the total number of instances available for training. A performance curve for three problems in the 20 Newsgroups data set [29] is shown in Figure 1. In this data set, the classes have nearly the same number of positive instances (1000 or near 1000 per class). For the concepts – *graphics* and *ms-windows.misc*, the learner achieves the maximum attainable accuracy (0.70 F1) after seeing about 2048 (2^{11}) examples. The value 2048 can be considered to be an overestimate on \hat{i} . For *sci.crypt*, the learner achieves its peak after seeing about 1024 examples, making it a somewhat easier concept (by our definition of complexity) than the other two. Each instance is chosen with the expectation that adding it to the training set will improve accuracy significantly. Since at each stage we are adding an example based on an estimate of its value to the training set, the complexity measure is approximate. We can tighten the estimate by providing the learning algorithm with as much information as possible: for example, a large pool may help. The advantage of using active learning is that the classifier needs to be trained only $O(M)$ times. How close this estimated complexity is to the true complexity is dependent on the ability of the learner to leave out redundant instances in its training.

A keen observer will note that the rate at which the performance improves of the *ms-windows.misc* is initially higher than that of *comp.graphics*. It seems intuitive that *ms-windows.misc* should be considered to be less complex than *comp.graphics*. Our first approximation of the minimum number of instances does not capture this aspect of learning. We factor in the learning profile by

⁵Regularization can be viewed as effectively a soft method for feature subset selection (e.g., see [41] on L1 regularization.).

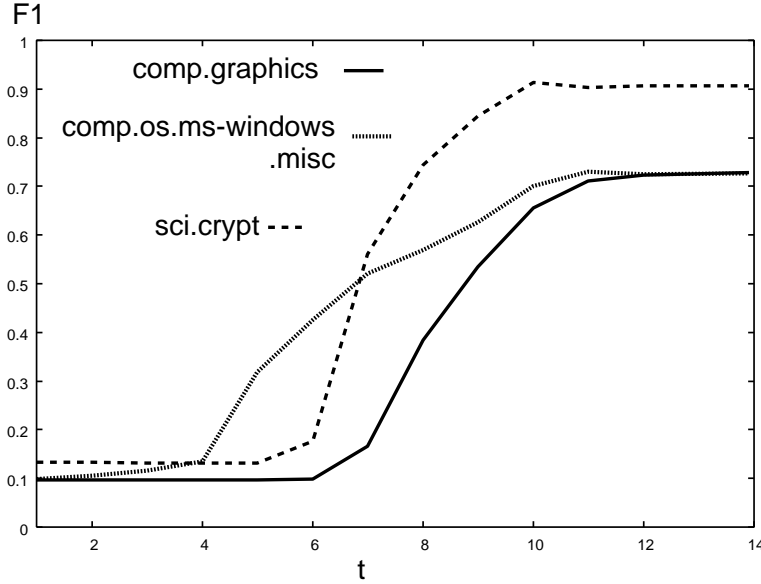


Figure 1: Learning curves for a single classifier on three problems. The number of actively picked instances is 2^t , and F1 (Y-axis) is the harmonic mean of precision and recall.

considering the area under the learning curve computed as

$$\text{AUC}_{\log} = \sum_{t=1}^{\log_2 M} F1(2^t, N)$$

We measure performance at exponentially increasing intervals, and compute the area under the learning curve, plotted with a logarithmic X-axis. AUC_{\log} implicitly gives a higher score to problems that converge more rapidly in the early stage of learning than later. To obtain a quantity that measures the profile of learning, we define the *active learning convergence profile* as follows:

$$p_{\text{al}} = \frac{\sum_{t=1}^{\log_2 M} F1(2^t, N)}{\log_2 M \times F1(M, N)} \quad (1)$$

p_{al} is the area under the normalized active learning curve (See Figure 2(a)), with a range between 0 and 1 and is independent of the number of instances needed for learning. Higher p_{al} implies faster convergence. The p_{al} values for the three problems in Figure 1 – *ms-windows.misc*, *comp.graphics* and *sci.crypt* are 0.61, 0.45 and 0.55, respectively. Note that even though the maximum accuracy achieved for *sci.crypt* is much higher (0.90 F1) than for the other two problems, the rate of active learning of *sci.crypt* is more similar to *comp.graphics*. The concept *ms-windows.misc* has the best rate of learning in the early stages. All these properties are captured by the p_{al} values.

We now describe the instance complexity measures developed using the approximations to \hat{i} and p_{al} . For all measures, a higher value of complexity implies a more difficult problem.

1. Instance size complexity (n_i) is the logarithm (base 2) of the number of instances needed to achieve 95% of the best performance (when all features are available). We expect that, given a desired accuracy level is determined (in our case, 95% maximum reachable accuracy), the empirically computed n_i is an over estimate on $\log_2(\hat{i})$ and the tightness of it depends on the active learning algorithm. We chose a threshold of 95%, rather than waiting for the curve to reach its peak, with the hope of capturing the point where most of the concept is learned. It is easier to check for an accuracy threshold near but below the maximum accuracy. Furthermore, the rate of improvement at the final stages of learning, before the concept is fully learned, can be very slow [42] with possibly several thousand instances contributing to a tiny improvement in performance, unnecessarily inflating the complexity score (See Figure 2(a)).

2. Instance profile complexity (I_{pc}) is given as $I_{pc} = 1 - p_{al}$, and we have $0 \leq I_{pc} \leq 1$. The active learning curve and hence the value of I_{pc} obtained is again subject to the active learning algorithm and will be less than the ideal case (theoretical best subset of instances). Therefore I_{pc} is an overestimate on the true profile complexity.

3. Combined instance complexity ($C_i = I_{pc} * n_i$) is the product of I_{pc} and n_i . n_i is an approximation of $\log_2(\hat{i})$, but does not include the rate of learning. I_{pc} includes only the rate of learning and does not contain information about the number of instances needed to achieve the best performance. C_i , a measure that is a fraction of n_i , incorporates both the learning profile and the sample size required.

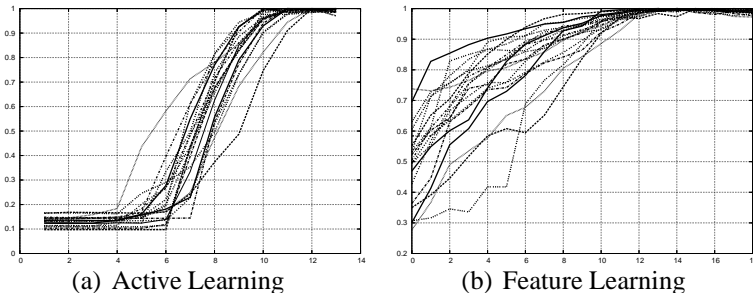


Figure 2: Normalized learning curves (active learning and feature learning) for 20 Newsgroups.

3.2 Feature Complexity Measures

Our feature complexity measures mirror our instance complexity measures, and are designed to capture the complexity of the problem in terms of the number of features needed to reach the best possible performance, when all the training instances are available. Again, instead of evaluating 2^N combinations of features, we estimate an approximation of the true feature complexity by

an intelligent ranking of the features in the order of decreasing discriminative ability for a given classification problem. The ranking procedure uses a large number of training documents and a feature selection criterion, such as the information gain [56]. We consider the performance of the classifier constructed using k top-ranking features. We plot a feature learning curve by plotting performance at exponentially increasing intervals of k . The normalized area under this feature learning curve, *the feature learning convergence profile*, p_{fl} is computed as follows:

$$p_{fl} = \frac{\sum_{t=1}^{\log_2 N} F1(M, 2^t)}{\log_2 N \times F1(M, N)} \quad (2)$$

Normalized feature learning curves for the 20 Newsgroups corpus are shown in Figure 2(b). Our feature complexity measures are

1. Feature size complexity (n_f) is the logarithm (base 2) of the number of features needed to achieve 95% of the best performance (when all instances are available).

2. Feature profile complexity (F_{pc}) is defined as $F_{pc} = 1 - p_{fl}$, thus $0 \leq F_{pc} \leq 1$. The computed value of F_{pc} is limited by the accuracy of the feature selection algorithm.

3. Combined Feature complexity (C_f) is a measure similar to C_i , and is defined as $C_f = F_{pc} * n_f$. How good the estimate of the true feature complexity obtained this way is dependent on the feature selection algorithm used.

4 Methods

We first describe the two linear classifiers of choice: perceptrons and support vector machines. We then describe our choice of active learning methods, followed by our choice of feature selection techniques.

4.1 Classifiers

A linear classifier, learned via a robust learning method, is usually sufficient for best accuracy in text classification (compared to other existing methods) in part due to the very high dimensionality of text. A simple algorithm for a linear classifier is the mistake-driven **Perceptron** algorithm [48]. The perceptron learns a linear function of the form $f(X_i) = w \cdot X_i + b$. It is a mistake-driven, incremental algorithm: when a new training example is added, the weight vector is adjusted only if it is misclassified. Therefore, the classifier needs to be trained less than (or equal to) $|\mathcal{T}|$ times, where \mathcal{T} is the training set. The correction to the weight vector is a simple adjustment of the form $w_i = w_i + \eta Y_i X_i$, for every instance X_i that is misclassified. Y_i is the true label of X_i ($Y_i \in \{+1, -1\}$). The parameter η is called the learning rate.

SVMs are learning techniques that have gained much popularity [57], particularly so for text classification [25, 53]. For many linearly separable problems there can be more than one hyperplane that separates the data (see Figure 3(a)). An SVM on the other hand, is a maximum margin

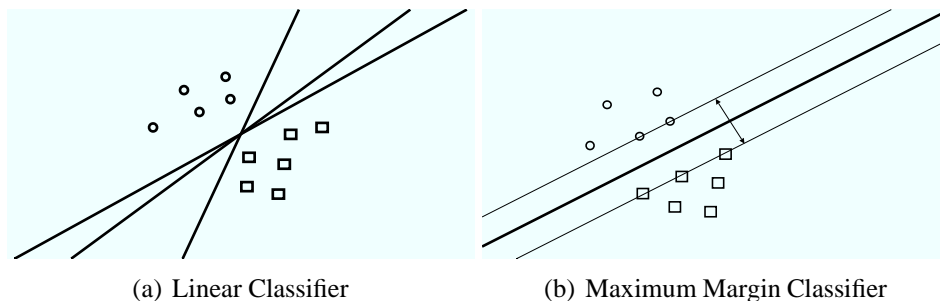


Figure 3: Linear classifiers.

classifier that tries to find the hyperplane that results in a maximal separation of the two classes (See Figure 3(b)). The margin is the distance between the positive example closest to the hyperplane and the negative example closest to the hyperplane, with the distance being measured along a line perpendicular to the hyperplane. Although the motive for a margin that is maximal seems intuitive, it is also well motivated by the Vapnik-Chervonenkis theory stating that such a hyperplane lowers expected test error [57]. Support vector machines have been observed to be effective in many domains, and especially so for text classification and filtering [25, 9]. We use a regularization parameter of $C = 1$ in our experiments.

4.2 Instance Selection

Any method for instance selection, such as random or uncertainty sampling (to be described), yields a sequence of instance subsets, growing in size, which allows us to compute the instance complexity measures as described in Section 3.1.

One method for instance selection is **uncertainty sampling** [32]. Uncertainty sampling is a simple, efficient, and commonly used type of active learning in which the example that the user (teacher) is queried on is the unlabeled instance that the classifier is least confident about. When the classifier is an SVM, unlabeled instances closest to the margin are chosen as queries [54]. If an uncertain instance lies exactly on the hyperplane it results in a reduction of the version space in exactly half [54]. If we can keep querying the user on examples that lie on the hyperplane we can decrease the number of training examples exponentially (by reducing the version space by half with each query) when compared to the case when the training data is obtained through random sampling. In reality, there may not be an example exactly on the hyperplane at each round of active learning. Or the problem may not be linearly separable and there can be label noise (e.g., mislabeled instances or outliers). Hence we do not see the theoretical exponential decrease, and different active learning methods perform differently on different problems.

One particular problem is that in many practical cases, the positive training instances are a tiny fraction of the whole population of instances, and thus the active learner may have difficulty

finding good near boundary instances (wherein a considerable portion of such instances need to be positive). In such cases, one prerequisite is that the pool from which the instances are examined and selected, in each active learning iteration, should be sufficiently large, so that, with high probability, the pool would contain at least several positive instances as well as negative instances near boundary (*i.e.*, basically useful instances).

Another aspect that affects the complexity scores is the choice of active learning (selective sampling) method. Some selective samplers may be better than others in certain cases. For instance, when there is class noise, uncertainty sampling may focus on outlier and uninformative instances, and thus waste learning resources. We briefly discuss the findings on active learning next.

For many text classification problems, uncertainty sampling has been observed to be significantly better than random sampling in accelerating learning [32, 54]. While many variations of selective sampling have been explored, for instance less myopic selection policies or using several active learners in tandem [34, 51, 54, 3], uncertainty sampling carries the advantages of simplicity and efficiency. Since uncertainty sampling is not ideal for every problem, we expect to obtain over estimates for our instance-based complexity measures. Note that since we take logarithms of the needed sample size, a superior method needs to beat uncertainty sampling by powers of 2 (in the ratio of sample sizes needed to reach near best performance) to yield an additive difference of 1 unit or more in the proposed estimate. Different choices of active learners can yield a significant relative size reduction on some problems, especially early in the training, *i.e.*, in achieving a good fraction of the maximum accuracy, say under 80%. Such differences can be important in practice. However, the relative performance compared to uncertainty sampling becomes closer as the accuracies reach say 90% of the maximum. See, for instance, the comparison plots and discussions for text classification [54, 51]. Thus, we expected that for this first study in the text classification domain, and with our rough measures of complexity, uncertainty sampling is adequate for our goals of measuring correlations among complexity measures, observing rough rankings of and trends in the difficulty of problems and data sets or domains. We will compare uncertainty sampling and random sampling, as subset selection methods, in our experiments.

Using uncertainty sampling with SVMs would involve retraining the SVM $O(M)$ times, which can be very time consuming as SVM training can involve quadratic optimization. Therefore, when we use SVM uncertainty sampling to compute p_{al} , we plot the learning curve only up to 1024 instances. To plot the complete active learning curve we use a another learning method – **a committee of perceptrons** (*e.g.*, [13]). The perceptron algorithm, being mistake-driven and online, takes less time than the SVM in each retraining, and perceptron committees, that we employ, can enjoy competitive accuracy performance (*e.g.*, [10, 37]). Of course, active learning using perceptrons may not be as effective as SVM uncertainty sampling and, in particular, the numbers that we obtain using different methods can be somewhat different. However, we find that the ranking of the problems by their complexity computed using the perceptron committee is almost identical to the ranking obtained using SVM uncertainty sampling on two data sets (refer to Figure 7).

4.3 Feature Selection

Any method for feature subset selection or feature ordering, such as information gain [40, 56], can yield a sequence of feature subsets, growing in size. This allows us to compute the feature complexity measures as described in Section 3.2.

Information gain is a simple, efficient, and commonly used measure in text classification for ranking features. It has been found to be quite effective in practical problems [46, 53, 18, 7]. Information gain is given as

$$IG = \sum_{c \in \{-1, +1\}} \sum_{\tau \in \{0, 1\}} P(c, \tau) \log \frac{P(c, \tau)}{P(c)P(\tau)}$$

where c denotes the class label (+1 or -1), and τ is 0 or 1 indicating the presence or absence of a feature, respectively.

Information gain is our primary feature selection method. However, information gain does not ignore redundant features, and more generally it does not address feature dependencies (a feature’s quality is computed independent of others). For instance, imagine that every important feature is replicated multiple times. Once one useful feature is selected, its copies become fully useless, but information gain would lead to picking all such features. Similar to the scenario with instance selection, this shortcoming can inflate our feature complexity scores. So we also experimented with **SVM LARS** [27], a forward selection technique when SVM learning is used. A forward selection technique is one wherein after each selection (or block of selections), the remaining features are re-evaluated and re-ordered. Given that it is a forward selection technique, LARS can skip redundant features in its feature selection, something information gain does not do. Therefore, we expect that LARS would capture the true feature complexity better. However, SVM LARS has a relatively high running time and we use it only in a limited way by computing p_{fl} by plotting the feature learning curve until the number of selected features gets to 1024 features. When we use information gain we are able to plot the entire learning curve.

Forward selection is a wrapper-based method [28]. It is also possible to use more costly versions of wrapper-based methods, involving, for instance, genetic search. In high-dimensional problems such as text, filtering methods, *i.e.*, simple ordering based on a certain criterion like information gain, are preferred because of their efficiency⁶ [60, 18].

4.4 Computing Accuracy Performance

Each time we compute $F1(2^t, 2^k)$ in equations 1 and 2, our aim is to find the best possible performance with a classifier trained on 2^t examples and 2^k features. We hope that by using active

⁶In a wrapper-based method, there is also an increased danger of overfitting or diminishing returns in terms of the generalization accuracy, if a large number of feature subsets is examined, and when the number of instances (relative to the number of features) is not sufficiently large.

learning with a large pool, and feature selection using a large training set, we obtain a fairly accurate estimate of this best classifier. The better the active learning and feature selection methods, the closer the estimate. Our experience with SVMs showed that with few training examples, much of the error is in a poor estimation of the threshold b . Other studies report similar observations (e.g., [61]). This also tends to be a problem with minority classes. Hence, to obtain a better and more estimate, we sweep through all values of b and use that b for which the F_1 is maximum on the held-out set. We call the maximum F1 achieved **MaxF1**. In fact in Table 2, the last column lists the MaxF1 values obtained with 90-10 training-test splits of the corpus. The complexity scores on various data sets are based on averaging the MaxF1 scores over 10 random 90-10 splits of each data set. That is, for each data set, and at each appropriate instance set or feature set size (t in equations 1 or 2), the MaxF1 score (on the held-out) is averaged over the 10 trials. Complexity scores such as I_{pc} and n_f are then computed. Note that the 10% held-out set in each trial is never used for instance selection or training.

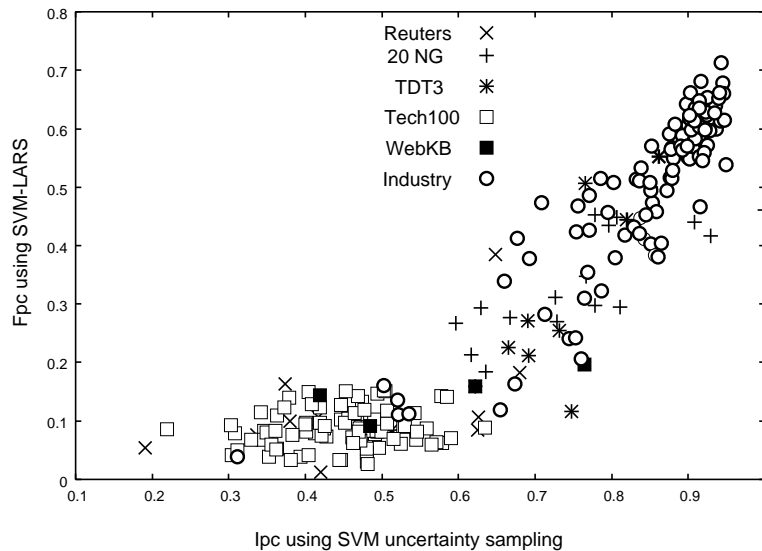


Figure 4: Correlation between I_{pc} and F_{pc} using SVM and LARS. Correlation of instance complexity and feature complexity is independent of methods used to compute the two.

5 Results

In this section and the next, we explore the utility of our measures. We describe the results of using our complexity measures on the 358 problems described in Table 2.

5.1 Correlation of Instance Complexity and Feature Complexity

Intuitively, good instance and feature complexity measures should correlate well: a problem that requires a large number of instances, tends to require finding the right mix of weights for a relatively large number of features, and vice versa. This intuition is related to theoretical results on the convergence (number of mistakes) of certain feature-efficient online algorithms, such as Winnow. For Winnow, the number of mistakes during convergence to a good hyperplane is directly dependent on the number of relevant features, at least for learning disjunctions of such features [35], and only logarithmically dependent on the total number of features. On the other hand, our empirical measures are imperfect: both uncertainty sampling and feature ordering by information gain, while relatively efficient, have a number of shortcomings. There are also other factors, such as the total number of features, that will affect complexity. Therefore, we would like to find out the extent of the correlation that exists on actual problems (if any).

Figure 4 illustrates that I_{pc} and F_{pc} of problems, computed using SVM uncertainty sampling and LARS are highly correlated ($r = 0.95$).⁷ The plots of I_{pc} vs. F_{pc} computed using perceptron committees and information gain look similar, albeit with a slightly lower correlation coefficient ($r = 0.81$ ($p < 2.2e^{-16}$)). The SVM methods show higher correlation, probably because they have the same underlying SVM learning, and SVM LARS does a better job of feature ordering for the SVM learner than information gain does for perceptron. Additionally, n_i and n_f (computed using perceptron committees and information gain) are strongly correlated ($r = 0.613$ ($p < 2.2e^{-16}$)) and therefore C_i and C_f are also strongly correlated ($r = 0.682$ ($p < 2.2e^{-16}$)).

We also experimented with random sampling for instance selection. The table below shows the correlation coefficients for I_{pc} and F_{pc} for various combinations of classifiers, instance selection mechanisms, and feature selection mechanisms for these 6 corpora.

| classifier | Feature Sel. | Instance Sel. | r |
|------------|--------------|---------------|------|
| SVM | LARS | Active | 0.95 |
| SVM | LARS | Random | 0.88 |
| Perceptron | Info. Gain | Active | 0.81 |
| Perceptron | Info. Gain | Random | 0.79 |

That instance complexity (the minimum number of instances needed to learn a concept) and feature complexity (the minimum number of features needed to learn a concept) are highly correlated may not be surprising since both are probably related to the Kolmogorov complexity⁸ of the learning problem. That the complexity measures exhibit this strong correlation raises our confidence in the utility of these measures.

⁷ r is Pearson's correlation coefficient, and $r=1$ denotes perfect correlation

⁸The complexity of a string is measured by the length of the shortest universal Turing machine program that correctly reproduces the observed data. Remember that K-complexity is only theoretical and cannot be computed.

| Corpus | Instance Complexity Measures | | | Feature Complexity Measures | | |
|----------|------------------------------|--------------|-------------|-----------------------------|--------------|-------------|
| | I_{pc} | n_i | C_i | F_{pc} | n_f | C_f |
| Tech100 | 0.04 (0.06) | 3.24 (2.23) | 0.20 (0.33) | 0.07 (0.02) | 1.89 (1.43) | 0.14 (0.14) |
| WebKB | 0.31 (0.13) | 8.75 (0.50) | 2.72 (1.04) | 0.11 (0.04) | 4.00 (2.16) | 0.51 (0.47) |
| Reuters | 0.35 (0.13) | 8.20 (1.03) | 2.93 (1.24) | 0.12 (0.07) | 4.80 (2.04) | 0.69 (0.56) |
| BNC | 0.39 (0.16) | 7.93 (1.91) | 3.34 (1.73) | 0.24 (0.11) | 11.47 (3.83) | 2.97 (1.60) |
| Enron | 0.46 (0.09) | 8.33 (0.87) | 3.82 (0.94) | 0.13 (0.06) | 7.67 (4.42) | 1.18 (0.70) |
| 20NG | 0.48 (0.04) | 10.40 (0.68) | 5.04 (0.71) | 0.23 (0.08) | 10.05 (1.39) | 2.32 (0.95) |
| TDT3 | 0.48 (0.13) | 9.30 (1.06) | 4.55 (1.53) | 0.20 (0.04) | 6.50 (1.78) | 1.34 (0.53) |
| RCV1 | 0.53 (0.14) | 10.67 (1.84) | 5.81 (2.25) | 0.23 (0.09) | 7.69 (2.04) | 1.81 (0.79) |
| Industry | 0.59 (0.12) | 10.34 (1.43) | 6.20 (1.71) | 0.29 (0.09) | 5.97 (1.52) | 1.77 (0.61) |

Table 3: Difficulty measures for different corpora. The higher the value, the more complex the problem. Values in brackets indicate standard deviation. The complexity is computed using the perceptron algorithm with uncertainty sampling for instance selection and information gain for feature ordering and selection. The description of the corpora (and the problems we chose from each to compute complexities) is given in Section 2.

5.2 Difficulty of Domains and Problems

We now benchmark all 9 corpora as easy or difficult for active learning using our complexity measures. Table 3 shows the complexity of different data sets. By all measures the Tech100 data set ranks as the easiest, followed by WebKB and Reuters. BNC, Reuters-RCV1, 20 Newsgroups and the Industry sector corpora are difficult by both our instance complexity and feature complexity measures. This result is better illustrated in the chart in Figure 6. This figure reaffirms the high correlation between instance complexity and feature complexity. That most corpora have problems of varying difficulty is demonstrated by the standard deviation of the scores in Table 3. Even though the BNC corpus is small (less than 3000 documents) it falls into the difficult end of the spectrum, implying that genre classification is more difficult than subject-based categorization.

The rankings of corpora using F_{pc} computed using SVM with LARS and Perceptron with information gain are also nearly identical, as is illustrated by Figure 5 (We show only a subset of the problems to illustrate this, because of the slow running time of LARS). The rankings of individual problems in these two corpora using F_{pc} computed using these two methods also correlate fairly well ($r=0.73$). The F_{pc} scores for individual problems in the Reuters-21578 and 20 Newsgroups using both methods are illustrated in Figure 7. Our results also support previous results that say that 20-Newsgroups consists of problems that are more difficult than Reuters-21578 and that problems like *wheat* are much easier with lower feature complexity as compared to *acq* [5, 24].

The Tech100 data set is a result of the efforts of [14] to obtain a data set containing problems of varying difficulty in terms of maximum accuracy performance achievable. Yet, we find that all

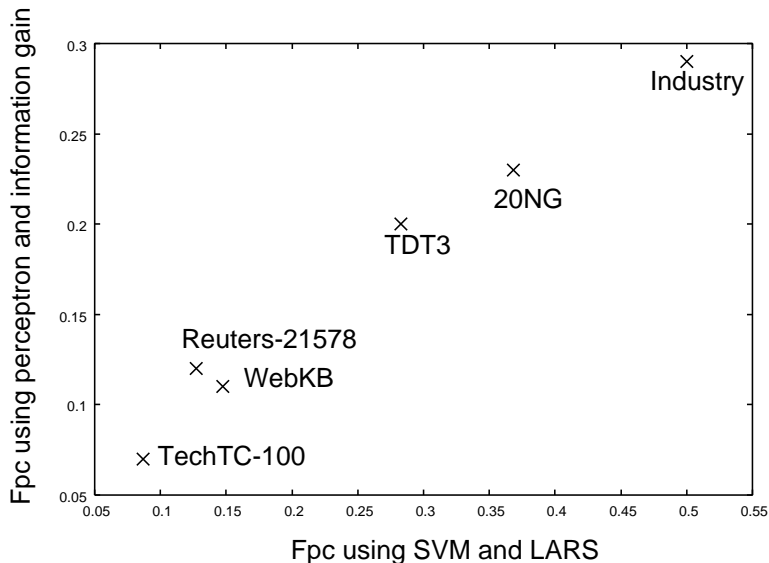


Figure 5: Ranking using F_{pc} computed by two different methods results in a similar ranking of corpora.

the problems in this data set are of low complexity, i.e., a few well-chosen examples or features are sufficient to achieve the optimal accuracy.

5.2.1 Number of Instances per Feature, and Learning Profiles

An interesting question is the number of useful instances that may be needed to reach adequate performance, *per useful feature*, on typical problems. Looking at the average feature complexity (n_f) and instance complexity (n_i) columns in Table 3, we find that the number of training instances required, *averaged per useful feature* (i.e., $2^{n_i - n_f}$), ranges from a handful or less to tens of instances. The WebKB and Industry data sets rank high based on this measure ($n_i - n_f > 4$), while for the BNC data set, $n_i - n_f < 0$. The shortcoming of information gain ordering (its failure to ignore redundant features, see Section 4.3) may in part explain this observation on the BNC data set.

As may be expected, the averages for instance and feature learning profile complexities are mostly close to or below 0.5 (I_{pc} and F_{pc}), meaning that most of the gains in accuracy are obtained relatively early in learning. Recall, however, that for computing area under the curve, we use a logarithmic x-axis (the number of instances or features doubles). We observe that the instance profile complexity averages tend to be higher than the feature profile complexities: learning is slower as a function of increasing size of training instances (when all the features are available) compared to accuracy improvements when increasing size of the relevant features (when all the instances are available). In fact, for half of the data sets, the average instance profile complexities are near 0.5 or exceed it.

5.2.2 Dependence of Complexity on the Corpus and the Choice of Classes

The TDT corpus consists of English newswire documents (Eng News), the output of an automatic speech recognizer system for English broadcast sources (Eng ASR), machine-translated newswire sources (MT News), and broadcast sources in Mandarin preprocessed through an automatic speech recognition system and a machine translation system (MT ASR). We measured the difficulty of each of the subsections of this corpus. The C_f values for event-based categorization are shown in the second column of Table 4.

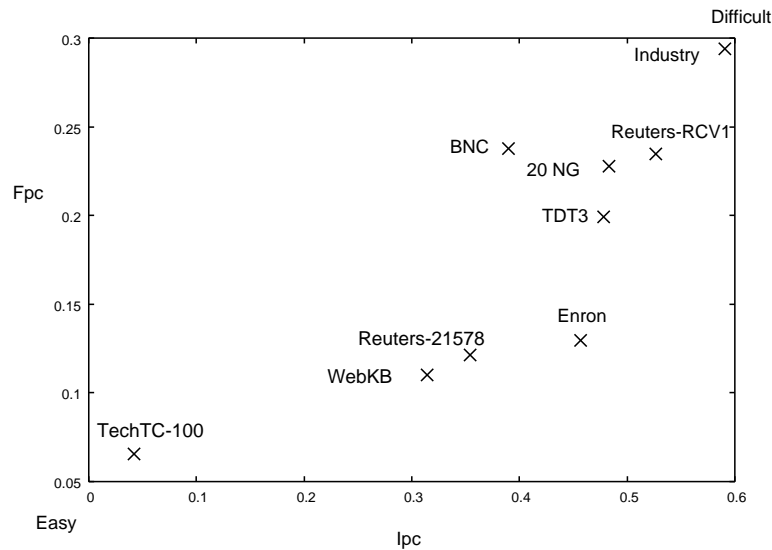


Figure 6: Instance Complexity (I_{pc}) and Feature Complexity (F_{pc}). A higher value of complexity indicates a difficult problem. Notice how instance complexity and feature complexity are correlated.

The English subsection of the corpus is easier than the machine-translated one, which is more noisy. For example, topic 30036 is *Nobel Prizes Awarded*. The feature complexity of this problem in each subset is shown in the third column. The most important words in English newswire and English ASR are (as expected) *Nobel*, *prize*, *Saramago* (person who won it), and so on, making classification in English newswire (Eng News) relatively easy. However, in the machine-translated output (MT News and MT ASR) the most important keywords are *promises*, *Bell*, *prize*, and *award*. The word *Nobel* is consistently translated to *promises Bell* in documents whose original source is *Mandarin*.⁹ Names like *Saramago* that are highly discriminatory in English are out of vocabulary in the machine translated documents, making the classification problem even harder. In addition, a multi-source setting (newswire, broadcast and multiple languages) can be more difficult than

⁹Nobel is a three-character word in Mandarin, the first of two of which also correspond to the English word *promises* and the third of which corresponds to the English name *Bell*.

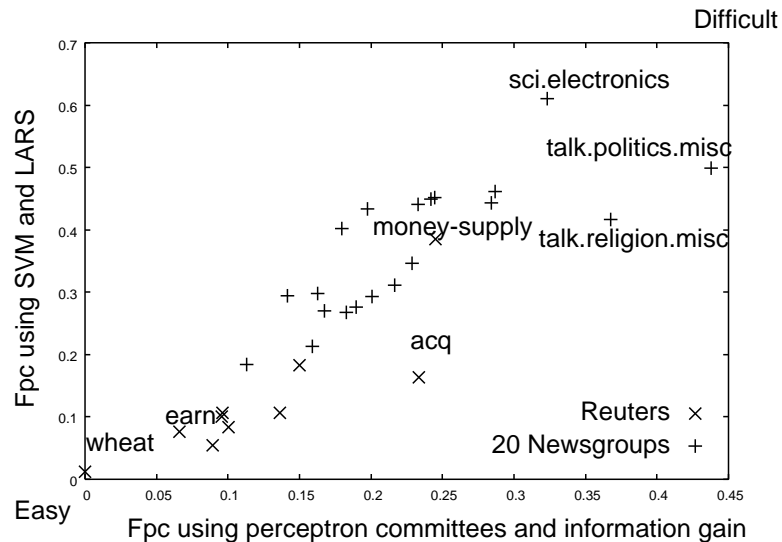


Figure 7: Feature complexity (F_{pc}) scores of problems in the Reuters-21578 and 20 Newsgroups corpora computed using two different methods. The higher the complexity, the more difficult the problem.

considering each source alone, as the vocabulary across sources differs depending on the MT and the ASR systems used.

| Subset of TDT3 | C_f by class type | | | |
|----------------|---------------------|----------------------|---------|-----------------------------------|
| | Events | <i>Nobel Awarded</i> | Subject | <i>Legal & Criminal cases</i> |
| Eng News | 0.65 | 0.27 | 2.03 | 2.56 |
| Eng ASR | 0.95 | 0.14 | 2.02 | 2.78 |
| MT News | 1.38 | 3.25 | 2.12 | 2.61 |
| MT ASR | 1.22 | 3.48 | 1.50 | 2.03 |
| Whole corpus | 1.34 | 1.60 | 2.78 | 3.30 |

Table 4: Difficulty of the TDT corpus when broken down by source and by category type.

So far we have considered categories based on events in the TDT corpus, and *Hurricane Mitch* and *Hurricane George* were different categories. The TDT corpus is also annotated by broader subjects like *natural disasters* and *elections*, the feature complexity of which is given in the column titled 'Subject' in Table 4. The last column shows the C_f values for an example topic – *legal and criminal cases*. The important features for classifying by subject are words like *court* and *law*, which do not suffer from as many ASR and MT errors, making the difficulty of subject-based

classification about the same in each source type, and even in the whole corpus (see the Subject column of Table 4).

We observe that Reuters-21578 comes in the range of less complex corpora and RCV1 in a higher complexity range, even though both corpora are composed of Reuters news stories. The Reuters-21578 categories come from the Construe system [21], and RCV1 categories are derived from a rich taxonomy derived for the Reuters Business Briefing [31]. Some of the documents in RCV1 were automatically assigned to their categories using a descendant of the Construe system. We found two topics that were common to both corpora, viz., *earnings* and *acquisitions*. Each of these two topics had F_{pc} values of 0.09 and 0.23 in Reuters-21578 and F_{pc} values of 0.154 and 0.192 in RCV1. In Reuters-21578 *earnings* has a rank of 3 out of the 10 topics when sorted in increasing order of difficulty, and in RCV1 it is the 13th of 87. Likewise, *Acquisitions* is the 9th topic in Reuters-21578 and the 51st in RCV1. So *acquisitions* in Reuters-21578 is much harder relative to the other topics than the same topic in RCV1. The difficulty of a topic depends not only on the size of the corpus, but also on how "confusable" the other topics are with it. The hierarchy in RCV1 may add to the complexity where nodes under the same parent may be more difficult to separate. In addition, RCV1 has a semi-automatic labeling process, which may be noisier and may add to its complexity.

5.2.3 Limitations

In summary, we note that the same corpus can exhibit different complexity levels and complexity ranges depending on the types of classes defined on it. Moreover, the "same topic", such as "Baseball", can exhibit different complexities for learning depending on the corpus or part of the corpus in which instances are annotated for it. The differences depend on several factors, including the set of features utilized, corpus size, number of classes defined on the corpus, and how similar those classes are (thus how hard it is to distinguish among them). Our ranking of the corpora should be understood to be relative to the classes we selected (for instance, RCV1 also has classification by the type of industry, which we did not use).

We also caution that our complexities are computed with respect to the many choices we made. The choices include the learning algorithm, the selective sampling and the feature selection procedures, the categories picked for each data set,¹⁰ the accuracy measure, and the type of features. Changing the choices will affect the difficulty scores and possibly the rankings. Furthermore, as the methods are imperfect, they most likely exhibit different degrees of approximation on different problems: on some problems or data sets, the scores may be very close to the lowest possible complexities, while on others they can be substantially off. Our rankings should be understood with these limitations in mind (see also Section 7.1).

¹⁰For some domains, we picked fewer classes than available, as described in Section 2.

6 Implications for Human-in-the-Loop Learning

The dual nature of complexity seems to imply that an intelligently picked feature can be as good as or better than an intelligently picked instance. This means (in theory at least) that we can actively learn by intelligently picking and weighting features. Of course, labeling features may not be cognitively as easy as labeling instances. A human, with sufficient knowledge of a category, would be able to effectively label many instances with category labels (some instances such as fuzzy ones can make labeling difficult), whereas labeling the features (with category labels or even merely indicating their relevance) may be harder. Relevance of a feature can depend on factors such as the corpus and the learning algorithm used. For example, it is not easy to determine whether the feature *drivers* is relevant in discriminating between *comp.graphics* and *ms-windows.misc*. Our initial guess was that humans may be able to judge a few features fairly quickly, and that labeling these few features would be equivalent to labeling a handful of documents, but the latter would be more time-consuming. Our preliminary experiments showed us that labeling a feature is more than five times faster than labeling an instance. We found that users can pick the most predictive features fairly accurately [45]. For problems of very low feature complexity, learning may be stopped once features are picked. For medium complexity problems, the user may need to mark several instances in addition to marking the features to achieve an acceptable level of accuracy. For very complex problems, feature selection may be much more difficult for the user, and instance feedback is the more reasonable alternative. Hence, we think a tandem approach of asking on instance feedback and feature feedback can be very beneficial: if the problem is of low to medium complexity, a few features that the user marks will quickly lead the classifier to convergence; if the problem is of high complexity, the user would not be able to recommend features (they may not be obvious) but can provide feedback on instances instead. In Section 6.1 we show evidence for this hypothesis. In particular, we observe that feature feedback accelerates active learning by an amount that is highly inversely correlated with the feature complexity of the problem.

6.1 Experiments

We saw that instance complexity and feature complexity are two sides of the same coin – a problem for which a few intelligently chosen instances can be used to build a good classifier tends to be one for which a few good features are very good predictors of class membership. In past work, we have observed that users can identify the most relevant features with reasonable accuracy [45]. The same work found that, on the problems tested, labeling features is about 5 times faster than labeling documents. From these results we hypothesize that coupling intelligent feature selection with intelligent document selection should accelerate active learning. Asking users to come up with features a priori is quite difficult. Users found it difficult to determine the relevance of a feature, without having seen any relevant documents. Hence an interleaved approach of asking the users to mark relevant features in tandem with documents that they label is probably cognitively easier. In subsequent work we have found that native English speakers could fairly easily point out ma-

chine translation errors of the kind discussed earlier where "Nobel" was consistently erroneously translated as "promises bell" [43]. That feedback significantly improved system performance.

In our previous work we built an active learning system for simultaneous document and feature feedback and showed that this dual feedback mechanism results in a much faster learning rate than traditional uncertainty sampling using an SVM as described in Section 3 [45]. In our InterActive Feature Selection algorithm, each time a document was picked by uncertainty sampling, the user was also asked to label 10 candidate features. The candidates were obtained by ranking the features by their information gain scores on the current labeled set, where the labels asked on features were *relevant* (is the feature discriminatory) or *non-relevant/don't know*. The labeled features were incorporated into the learning by scaling the value of that feature in all the instances. User feature feedback was simulated using an "oracle", the details of which can be found in [43]. We found that actual users could emulate the oracle to an extent that resulted in as much improvement as can be achieved using the oracle.

The active learning convergence profile p_{al} measures the rate of performance improvement or the speed of learning. We measured the speed of traditional uncertainty sampling (document feedback only) and that of the InterActive feature selection algorithm for all 358 problems benchmarked in this work. We measure performance only up to $T = 42$ labeled examples and plot the active learning convergence profile (p_{al} , refer Equation 1). Similarly, we measure p_{ifs} as the interactive feature selection convergence profile. Figure 8 plots the quantity $p_{ifs} - p_{al}$ for all 358 problems benchmarked in this work. The improvement in speed due to the incorporation of term feedback in addition to document feedback is inversely related to feature complexity as seen in Figure 8 ($r = -0.65$). Speed is improved by about 57% on average.

The *faculty* class in WebKB shows significant improvement in speed (see Figure 8). For this problem, the keywords *faculty* and *professor* are sufficient to obtain 93% of the maximum achievable accuracy (90.05% F1). Both these terms appear for feature feedback within the first five iterations in all 30 trials. Similarly, for the Enron corpus, one of the folders is almost completely classified by the sender of the e-mail, *Wilson Shona* (some other folders also contain some e-mails by *Wilson Shona*). The algorithm recommends his e-mail id for feedback in the early iterations, resulting in significant improvements in performance. The *miscellaneous* category in the BNC corpus does not gain from term feedback whereas *arts/cultural material* does, because of discriminatory keywords like *opera*, *actor*, or *theater* in the latter category, which when marked relevant improve performance significantly. There are a couple of outliers like the RCV1 category *reserves*, for which speed decreases by a large amount when term feedback is included. This may have occurred because a fixed scaling factor of 10 for the selected features is used to bias the linear classifier (SVM) learning algorithm. This may not be appropriate for every problem. An interesting question is whether there are more robust methods for asking and taking feature feedback into account.

We also note that some problems can be too easy (very low feature complexity), and simple active learning and possibly even plain random sampling (if class skew is not an issue) may have adequate speed.

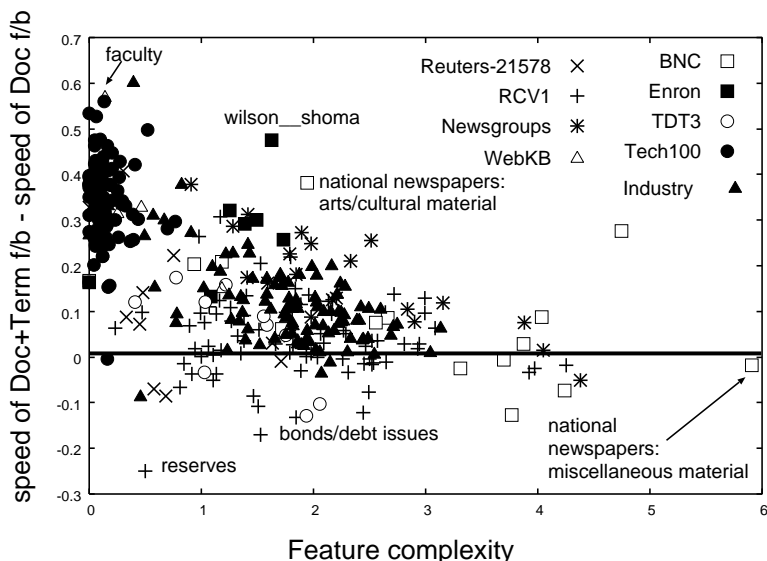


Figure 8: Difference in speed of active learning and InterActive Feature Selection as a function of complexity (C_f).

We report the performance (F1) for 9 corpora in Table 6.1, after 12 and 32 rounds of document feedback using traditional active learning and 12 rounds of interactive feature selection. Interactive feature selection always improves performance over active learning with only 12 documents. It is significantly better than 32 actively sampled documents for five of eight cases. The categories in the BNC corpus are by genre and this result can be interpreted quite intuitively: for categories like "prose" and "poetry" even intuitively it does not seem like any keywords can capture these concepts.

In this section we used our difficulty measures to better understand situations when feature feedback methods during active learning might be especially useful. We observed that, in linear classifier learning problems, feature feedback accelerates active learning by an amount that is inversely related to the feature complexity of the problem. For low- to mid-range feature complexity problems, revealing to the learning algorithm those features that are most important can focus the active learning better in the selection of documents to label and in computing the weights for features that matter most. In such cases, a few training documents combined with feature feedback can result in substantial improvement in accuracy with little additional labeled data. We find that many problems in our 9 corpora fall in a low to medium range of complexity ($0 < C_f < 2$), and stand to gain from such a dual feedback framework. Automated email foldering is an example domain that may have many relatively low complexity problems.

Future work in this direction includes using these or similar measures to explain other observations, such as when semi-supervised techniques may or may not work well. It may also be fruitful to explore methods for predicting the expected difficulty of a learning problem at the beginning

| Corpus | C_f | Only docs (Active) | | IFS |
|----------|-------|-----------------------|---------|--------------|
| | | 12 docs | 32 docs | |
| Tech100 | 0.20 | 0.486 | 0.594 | 0.847 |
| WebKB | 0.51 | 0.262 | 0.424 | 0.520 |
| Reuters | 0.69 | 0.516 | 0.570 | 0.651 |
| Enron | 1.18 | 0.218 | 0.444 | 0.465 |
| TDT3 | 1.34 | 0.202 | 0.259 | 0.336 |
| Industry | 1.77 | 0.071 | 0.123 | 0.199 |
| RCV1 | 1.81 | 0.134 | 0.260 | <i>0.231</i> |
| 20 NG | 2.32 | 0.180 | 0.259 | 0.336 |
| BNC | 2.97 | 0.209 | 0.332 | <i>0.264</i> |

Table 5: Improvement in F1 for corpora of different levels of difficulty. Numbers in bold indicate that InterActive Feature Selection is significantly better than when only documents are used for feedback. Numbers in italics indicate significantly lower performance than the case when $T = 32$.

stages of training (when limited labeled data is available). This can inform the subsequent learning strategy taken.

7 Related Work and Discussion

The classic “curse of dimensionality” informally states that the higher the dimension of the problem, the harder the problem (in this case, learning). However, our work goes beyond that and tries to measure the inherent complexity of the problem. A large dimensional learning problem may be easy if only a few features are required for learning it. We show here that actively picked examples reveal the complexity better, and we relate this to measures of feature complexity as well.

Ho and Basu [22] defined a set of measures that captured the complexity of the geometry of the boundary for a few artificial and real binary classification problems of low dimensionality. In comparison, our work is in the domain of text classification, where a linear hyperplane is often effective in making the geometry of the boundary less of an issue. We experimented with one of their measures of feature complexity, maximum Fisher discriminant ratio, to find that it did not correlate as well with I_{pc} ($r = 0.2$). We also measured how F_{pc} correlated with maximum accuracy and found the correlation to be not very high ($r=0.4$).

The difficulty in domains such as text is that relatively large amounts of training data (hundreds or thousands) may be needed to converge to the optimal hyperplane. We note that linear classifiers do well on a number of challenging high-dimensional real-world problems, for example, in natural

language and vision (e.g., [49, 50]). For other domains where active learning is used (e.g., [55]) but where the classifier is not linear it is less clear whether our complexity measures can directly be used. In our approach, to measure the (instance or feature) complexity of a problem, given the accuracy achieved on the whole training set is deemed acceptable, or given a certain accuracy level of interest, the crucial aspect becomes the techniques used for feature subset selection or instance subset selection. If the selection methods are effective in approximately capturing the number of instances and features required, the complexity scores obtained would be adequate. The goal is to say, for example, that two problems that have 9 and 10 instance complexities (n_i) are somewhat similar in difficulty, while a problem with instance complexity of 6 is significantly easier.

In our experiments, we made the assumption that near best accuracy is achieved when all the features are available to the learner, when robust learners such as well-regularized SVMs are used (and when training data is substantial). In these settings, feature selection is primarily used to reduce the training time or the space complexity of the learning system (or both). For instance, [18] performs an extensive study of feature ordering methods using linear SVMs, and the performance plots show that the optimal accuracies (F-measure) achieved are the same or close (less than 5% difference overall) compared to when all the features are used for almost all feature ordering methods that were tried, in particular information gain. Only for one ordering method, the proposed BNS method, the performance dips somewhat when all the features are used for some problems. Forman concludes that it is difficult to beat the performance of SVM trained on all the available features. There were cases for which the proposed BNS feature ordering could at times beat learning under all the features, in particular for high class skews (minority classes) [18].¹¹ In later work, it was observed that with improved vector representation (feature value scaling using BNS), feature subset selection did not improve accuracy [19]. We note that for some other learning methods, such as Naive Bayes or nearest neighbors, explicit feature selection can substantially improve accuracy (e.g., [60]), although it was found later that well-regularized SVMs are superior and may not require explicit feature selection for accuracy improvements, unlike nearest neighbors and Naive Bayes (e.g., [59]). As mentioned previously, in our experiments, we used n-grams as features (up to 3-grams) only if accuracy improved, and dropped rare features (with frequency below 5). There are other feature selection methods, such as wrapper-based methods (e.g., [28]). In text, with very high dimensionality, such methods tend to become too costly, and *filter*-based methods are preferred (e.g., [18, 60]).

Debole and Sebastiani [16] studied the difficulty of three versions of the Reuters-21578 corpus that are commonly used in publications. Their focus was on measuring accuracy, in particular macro and micro accuracy measures. They found that the version we used (smallest, including only top 10 classes) was somewhat easier in micro-average accuracy performance than the versions that included approximately 90 or more classes (but with many relatively small classes). [14] devel-

¹¹In that work, experimentation with different regularization parameters for SVM training was not performed (but the findings roughly remain the same when using different regularization parameters as well (private communication with the author)). We also note that we report on MaxF1 rather than F1, to lower the effects of poor threshold selection, especially problematic with few instances or minority classes.

oped a benchmark data set consisting of 100 text-classification problems with varying difficulty (accuracy ranging from 0.6 to 0.92). They also developed measures for predicting the difficulty of a problem, but this was in terms of its accuracy. Instead, our focus is on understanding how many features or examples are needed to achieve the maximum accuracy. In fact their data set, Tech-100, is the easiest data set according to our feature and instance complexity measures, and illustrates the fact that difficulty in terms of accuracy value is different from difficulty in terms of sample size or feature size requirements.

Previous work has noted the “low feature complexity” of problems in some of the commonly used data sets (e.g., [5, 24, 23]). For instance, [23] observes that many problems in commonly used UCI (low-dimensional) data sets, at that time, can be captured by simple rules. Our work is an attempt to quantify past observations, in particular in high-dimensional problems such as text. We investigate both feature complexity and instance complexity, and their relationship, and benchmark many standard text classification data sets. As our measures tend to be over estimates, given that the accuracy levels reached are acceptable, our conclusions have a one sided aspect: problems that we find relatively easy are easy, while problems that we find difficult may turn out to be easy if other methods are used.

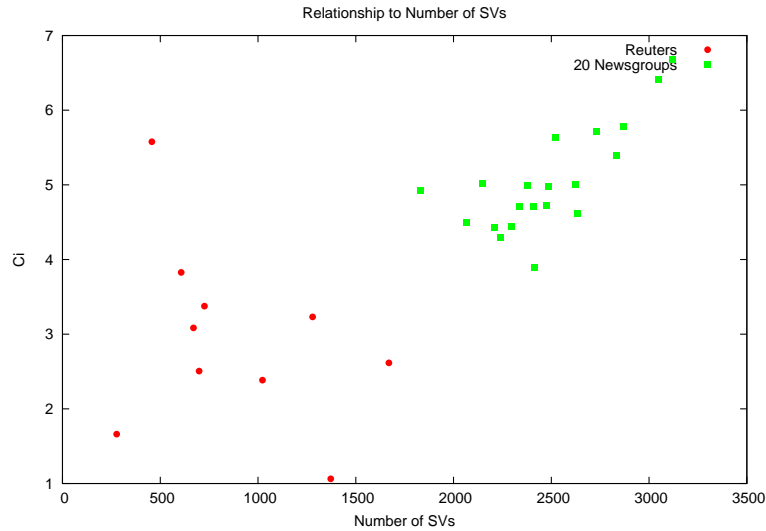


Figure 9: Relationship of C_i to the number of support vectors.

Other measures of complexity may exhibit similar properties. Gabrilovich and Markovitch [20] defined a feature complexity measure called *outlier count* that attempts to capture the number of important features for a given learning problem. The authors used outlier count to characterize problems for which decision trees are more accurate than SVMs, the latter being the main thrust of their work. The work here on the other hand is an in-depth analysis of complexity – both feature and instance. We experimented with outlier count, finding that it correlates with instance com-

plexity (I_{pc}) reasonably well ($r=0.610$), as do our feature complexity measures. Another candidate measure that might correlate well with instance or feature complexity is the number of support vectors when SVM learning is used. One advantage of this measure is efficiency, as only a single training step is required for each problem, and very fast algorithms for linear SVM training have been developed recently (e.g., [11]). A potential shortcoming (in addition to being limited to SVM learning) is that the number of support vectors on the boundary to obtain the best accuracy may be unnecessarily high (e.g., because of nearly identical instances), while effective selective sampling may attain adequate accuracy with substantially fewer instances. Still, strong correlation may exist. We performed a small experiment on the newsgroup and Reuters-21578 data sets to compare the number of support vectors used in the linear SVM classifier on each classification problem to our instance complexity measures C_i and n_i . We find a strong correlation: $0.754, p < 0.05$, when C_i is correlated against the number of support vectors, as shown in Figure 9. n_i also had a strong correlation of 0.89 to the log number of support vectors, and 0.92 to the number of support vectors. Probably no method is best in every scenario, but it is promising to see that a number of measures have significant positive correlation with one another on several problems.

7.1 Other Factors

Capturing the exact underlying complexity relates to maximum compression of a given string and is intractable. Thus, the subject of this work was to explore the utility of our approximate and relative measures, which depends on a number of factors, including the choice of features and the learning algorithm, as well as our chosen instance and feature selection techniques (see also Section 5.2.3). We reported comparisons in the choice of the learning algorithm and instance selection and feature selection methods. In general, however, different learning algorithms may have close but still different accuracies, and the addition of a single feature or a class of features can render a very hard learning problem trivial.

There are other potentially relevant factors to complexity that we have not used, including the average length of documents, the size of the feature set (N), and the proportion of the positive documents (for a given class) in the corpus. With our use of active learning for instance selection and information gain for feature selection, and our focus on roughly measuring complexity, these aspects are probably less influential, although this requires further study.

We aimed to measure rough complexity, useful for tasks such as comparing and ranking problems. Furthermore, our complexity measures are based on the logarithmic scale, akin to the Richter scale. Intuitively, the same fixed absolute difference, say in the number of instances, loses its significance as the number of instances required for learning increases. The logarithmic scale captures increases in the *relative* size requirements, which we expect is more appropriate. We restricted ourselves to text classification and provided evidence that, within this boundary, the measures are useful.

Blum and Langley [6] provide good motivation in introduction to this work. They discuss the problem of selecting relevant examples and relevant features as two ways of gathering relevant

information in a data set. They formally define the relevance of features and examples, and suggest using relevance as a measure of complexity. Their work is, however, theoretical and their definitions apply for classes that can be completely described (i.e., 100 % accuracy is achieved) by some conjunction or disjunction of features. Real-world problems like text classification are not so simple and it is not clear how their measures may be used to quantify complexity for real-world problems. They conclude their paper by stating the following *empirical challenge*:

Feature selection and example selection are tasks that seem to be intimately related and we need more studies designed to help understand and quantify this relationship. Much of the empirical work on example selection has dealt with low-dimensional spaces, yet this approach clearly holds even greater potential for domains involving many irrelevant features. Resolving basic issues of this sort promises to keep the field of machine learning occupied for many years to come.

Our work addresses some of the questions raised in their paper. We define measures that can be computed fairly easily on real-world text classification problems where linear classifier learning algorithms are successful, and demonstrate that instance complexity and feature complexity are highly positively correlated on the problems we tested.

8 Summary

Designing adequate empirical measures of learning difficulty is a balancing act between efficiency and utility. Such measures are necessarily relative to the choices made, including the choice of learning algorithm(s) as well as feature or instance subset selection methods. The approach explored here is relatively simple, and in principle applicable to different learning problems. The methods used are relatively efficient, and we presented evidence that they are useful in the domain of text classification, when robust linear learning algorithms are used. We obtained rough but useful measures of difficulty, leading to a fairly consistent ranking of problems, and exhibiting explanatory power, for example, in explaining the extent of benefit from feature feedback during active learning. In particular, we observed a high positive correlation between the instance complexity and feature complexity measures, indicating that they approximate the inherent complexity well. We benchmarked 9 corpora and 358 problems and used these measures to gain insights on the relative difficulty of a variety of text classification problems and domains. Our measures also capture how difficulty can be different even within a corpus depending on the type of classes (say subject or event) that one is trying to learn. We found that problems with low to medium feature complexity stand to benefit most from feature feedback, and we see that many experimental problems, such as email categorization, fall in this range (Table 3 and Figure 8). This has encouraging implications for the use of active learning with feature feedback for real-world filtering and email classification problems.

We hope that our analyses and domain rankings would serve to inform future research, for example in selecting corpora and anticipating results. Future work includes extending these measures and exploring other factors that may help further explain difficulty and variations in learning performance. Candidate factors include the proportion of positive instances and the dependency patterns among the features. It would also be useful to study difficulty measures in other learning problems and domains.

We note that our measures serve primarily for understanding or explaining learning behavior, such as convergence. Currently, they cannot be used to predict complexity on a new problem with no or few labeled instances. Such *prediction* of complexity appears to be a difficult if not an unsolvable task, unless real-world problems turn out to satisfy helpful properties. Given that we do not know at the outset how to predict whether a concept is going to be easy or difficult, and assuming that the teacher has prior knowledge on features, a tandem learning approach that mixes both feature and instance feedback may be the best general approach for accelerated learning, especially in the early stage of learning.

Acknowledgments

Many thanks to George Forman for valuable input and pointers.

References

- [1] J. Allan. *Topic detection and tracking*. Kluwer Academic Publishers, 2002.
- [2] C. Apte, F. Damerau, and S. M. Weiss. Automated learning of decision rules for text categorization. *ACM Transactions on Information Systems*, 12(3):233–251, 1994.
- [3] Y. Baram, R. El-Yaniv, and K. Luz. Online choice of active learning algorithms. *Journal of Machine Learning Research*, 5:2004, 2003.
- [4] R. Bekkerman. Document classification on Enron email dataset. 2004.
- [5] R. Bekkerman, R. El-Yaniv, Y. Winter, and N. Tishby. On feature distributional clustering for text categorization. In *Proc. 24th Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR)*, pages 146–153, New York, NY, USA, 2001. ACM Press.
- [6] A. Blum and P. Langley. Selection of relevant features and examples in machine learning. *Artificial Intelligence*, 97(1-2):245–271, 1997.
- [7] J. Brank, M. Grobelnik, N. Milic-Frayling, and D. Mladenic. Feature selection using linear support vector machines. Technical report, Microsoft Research, 2002.

- [8] G. Burnage and D. Dunlop. Encoding the British national corpus. In *English Language Corpora: Design, Analysis and Exploitation, Papers from the 13th Intl. Conf. on English Language Research on Computerized Corpora*, 1992.
- [9] N. Cancedda, N. Cesa-Bianchi, A. Conconi, C. Gentile, C. Goutte, Y. Li, J. Renders, and A. Vinokourov. Kernel methods for document filtering. In *Proc. 11th Text Retrieval Conf. (TREC)*. Dept. of Commerce, NIST, 2002.
- [10] V. R. Carvalho and W. Cohen. Single pass online learning. In *Proc. ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining (KDD)*, 2006.
- [11] C. J. Hsieh K. W. Chang, C. J. Lin, S. S. Keerthi, and S. Sundararajan. A dual coordinate descent method for large-scale linear SVM. In *Proc. Intl. Conf. on Machine Learning (ICML)*, 2008.
- [12] M. Craven, D. DiPasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam, and S. Slattery. Learning to extract symbolic knowledge from the World Wide Web. In *Proc. 15th National Conf. on AI (AAAI)*, pages 509–516, Madison, US, 1998. AAAI Press, Menlo Park, US.
- [13] S. Dasgupta, A. T. Kalai, and C. Monteleoni. Analysis of perceptron-based active learning. In *Proc. Conf. on Computational Learning Theory (COLT)*, pages 249–263, 2005.
- [14] D. Davidov, E. Gabrilovich, and S. Markovitch. Parameterized generation of labeled datasets for text categorization based on a hierarchical directory. In *Proc. 27th Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR)*, pages 250–257, 2004.
- [15] A. Dayanik, D. D. Lewis, D. Madigan, V. Menkov, and A. Genkin. Constructing informative prior distributions from domain knowledge in text classification. In *Proc. 29th Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR)*, pages 493–500, New York, NY, USA, 2006. ACM Press.
- [16] F. Debole and F. Sebastiani. An analysis of the relative hardness of Reuters-21578 subsets. *Journal of the American Society for Information Science and Technology*, 56, 2004.
- [17] S. Dumais, J. Platt, D. Heckerman, and M. Sahami. Inductive learning algorithms and representations for text categorization. In *Proc. 8th Intl. Conf. on Information and Knowledge Management (CIKM)*, 1998.
- [18] G. Forman. An extensive empirical study of feature selection metrics for text classification. *Journal of Machine Learning Research*, 3:1289–1305, 2003.
- [19] G. Forman. BNS feature scaling: an improved representation over tf-idf for SVM text classification. In *Proc. Intl. Conf. on Information and Knowledge Management (CIKM)*, 2008.

- [20] E. Gabrilovich and S. Markovitch. Text categorization with many redundant features: Using aggressive feature selection to make SVMs competitive with $c4.5$. In *Proc. 21st Intl. Conf. on Machine Learning (ICML)*, pages 321–328, New York, NY, USA, 2004. ACM Press.
- [21] P. J. Hayes and S. P. Weinstein. CONSTRUE/TIS: A system for content-based indexing of a database of news stories. In *Proc. 2nd Conf. on Innovative Applications of AI (IAAI)*, pages 49–64. AAAI Press, 1991.
- [22] T. K. Ho and M. Basu. Complexity measures of supervised classification problems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(3):289–300, 2002.
- [23] R. C. Holte. Very simple classification rules perform well on most commonly used datasets. *Machine Learning*, 11(1):63–90, 1993.
- [24] T. Joachims. A probabilistic analysis of the rocchio algorithm with TFIDF for text categorization. In *Proc. Fourteenth Intl. Conf. on Machine Learning (ICML)*, pages 143–151, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc.
- [25] T. Joachims. Text categorization with support vector machines: learning with many relevant features. In *The 10th European Conf. on Machine Learning (ECML)*, pages 137–142, 1998.
- [26] M. J. Kearns and U. V. Vazirani. *An Introduction to Computational Learning Theory*. MIT Press, 1994.
- [27] S. Keerthi. Generalized LARS as an effective feature selection tool for text classification with SVMs. In *Proc. 22nd Intl. Conf. on Machine Learning (ICML)*, pages 417–424, August 2005.
- [28] R. Kohavi and G. H. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1-2):273–324, 1997.
- [29] K. Lang. Newsweeder: Learning to filter netnews. In *Proc. 12th Intl. Conf. on Machine Learning (ICML)*, pages 331–339, 1995.
- [30] D. Lewis. Reuters-21578 text categorization test collection, Readme (version 1.2). In <http://www.daviddlewis.com/resources/testcollections/reuters21578/>, 1997.
- [31] D. D. Lewis, Y. Yang, T. G. Rose, and F. Li. RCV1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5:361–397, 2004.
- [32] D. D. Lewis and J. Catlett. Heterogeneous uncertainty sampling for supervised learning. In *Proc. 11th Intl. Conf. on Machine Learning (ICML)*, pages 148–156, 1994.

- [33] F. Li and Y. Yang. A loss function analysis for classification methods in text categorization. In *Proc. 20th Intl. Conf. on Machine Learning (ICML)*, 2003.
- [34] M. Lindenbaum, S. Markovitch, and D. Rusakov. Selective sampling for nearest neighbor classifiers. In *16th National Conf. on AI (AAAI)*, 1999.
- [35] N. Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2(4):285–318, 1988.
- [36] N. Littlestone. Redundant noisy attributes, attribute errors, and linear-threshold learning using Winnow. In *COLT*, pages 147–156, 1991.
- [37] O. Madani and M. Connor. Large-scale many-class learning. In *SIAM Conference on Data Mining*, 2008.
- [38] A. McCallum, R. Rosenfeld, T. M. Mitchell, and A. Y. Ng. Improving text classification by shrinkage in a hierarchy of classes. In *Proc. 15th Int. Conf. on Machine Learning (ICML)*, pages 359–367, 1998.
- [39] A. K. McCallum. Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering. <http://www.cs.cmu.edu/~mccallum/bow>, 1996.
- [40] T. M. Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997.
- [41] A. Ng. Feature selection, L1 vs. L2 regularization, and rotational invariance. In *Proc. 21st Intl. Conf. on Machine learning (ICML)*, 2004.
- [42] F. Provost, D. Jensen, and T. Oates. Efficient progressive sampling. In *Proc. 15th ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining (KDD)*, pages 23–32, New York, NY, USA, 1999. ACM Press.
- [43] H. Raghavan and J. Allan. Passage feedback for news tracking. Technical Report IR-459, Center for Intelligent Information Retrieval, University of Massachusetts, Amherst, 2006.
- [44] H. Raghavan and J. Allan. An interactive algorithm for asking and incorporating feature feedback into support vector machines. In *Proc. 30th Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR)*, pages 79–86, New York, NY, USA, 2007. ACM Press.
- [45] H. Raghavan, O. Madani, and R. Jones. Interactive feature selection. In *Proceedings of IJCAI 05: The 19th International Joint Conference on Artificial Intelligence*, 2005.
- [46] H. Raghavan, O. Madani, and R. Jones. Active learning on both features and documents. *Journal of Machine Learning Research*, 7:1655–1686, August 2006.

- [47] R. Rifkin and A. Klautau. In defense of one-vs-all classification. *Journal of Machine Learning Research*, 5, 2004.
- [48] F. Rosenblatt. Two theorems of statistical separability in the perceptron. In *Symposium on the Mechanization of Thought Processes*, pages 421–456, 1959.
- [49] D. Roth. Learning in natural language. In *Proc. Intl. Joint Conference on AI (IJCAI)*, pages 898–904, 1999.
- [50] D. Roth, M. H. Yang, and N. Ahuja. Learning to recognize objects. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 724–731, 2000.
- [51] N. Roy and A. Mccallum. Toward optimal active learning through sampling estimation of error reduction. In *Proc. 18th Intl. Conf. on Machine Learning*, pages 441–448. Morgan Kaufmann, 2001.
- [52] R. Schapire, M. Rochery, M. Rahim, and N. Gupta. Incorporating prior knowledge into boosting. In *Proc. 19th Intl. Conf. on Machine Learning (ICML)*, pages 538–545, 2002.
- [53] F. Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47, 2002.
- [54] S. Tong and D. Koller. Support Vector Machine active learning with applications to text classification. *Journal of Machine Learning Research*, 2:45–66, 2002.
- [55] Simon Tong and Edward Chang. Support vector machine active learning for image retrieval. In *MULTIMEDIA '01: Proceedings of the ninth ACM international conference on Multimedia*, pages 107–118, New York, NY, USA, 2001. ACM Press.
- [56] C. J. Van Rijsbergen. *Information Retrieval, 2nd edition*. Dept. of Computer Science, University of Glasgow, 1979.
- [57] V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, 2nd edition, 1995.
- [58] X. Wu and R. Srihari. Incorporating prior knowledge with weighted margin support vector machines. In *Proc. Tenth 10th ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining (KDD)*, pages 326–333, 2004.
- [59] Y. Yang and X. Liu. A re-examination of text categorization methods. In *Proc. 22nd Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR)*, 1999.
- [60] Y. Yang and J. O. Pederson. Feature selection in statistical learning of text categorization. In *Proc. 14th Intl. Conf. of Machine Learning (ICML)*, 1997.

- [61] J. Zhang and Y. Yang. Robustness of regularized linear classification methods in text categorization. In *Proc. ACM SIGIR Intl. Conf. on Research and Development in Information Retrieval (SIGIR)*, 2003.
- [62] T. Zhang and F. J. Oles. Text categorization based on regularized linear classification methods. *Information Retrieval*, 4, 2001.